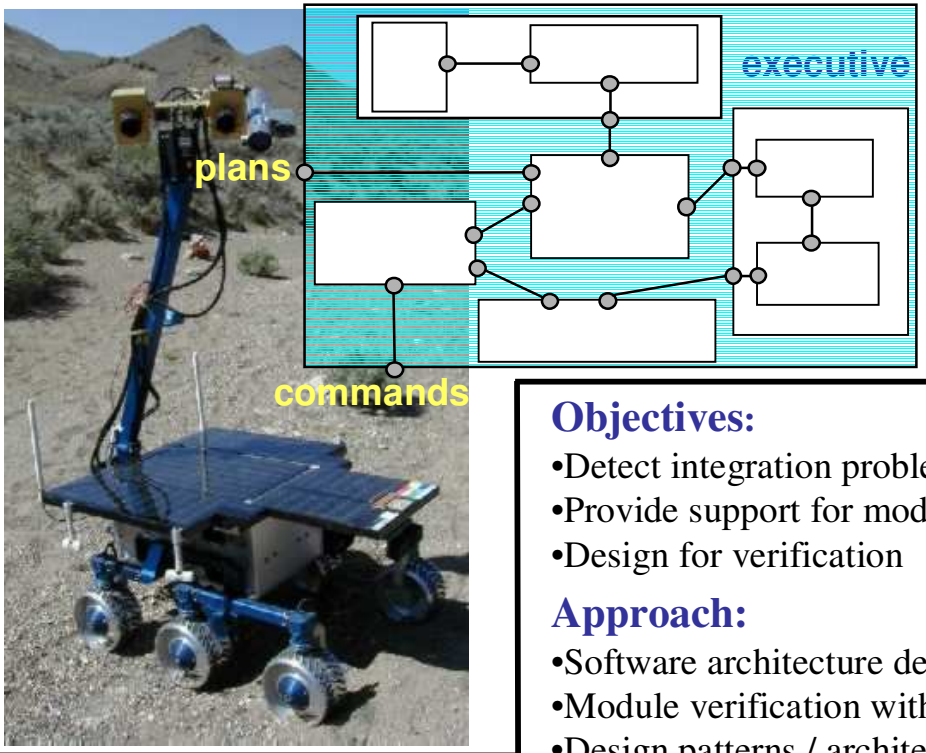
Modular Verification for Autonomous Systems



IS Relevance:

- Verification is essential for autonomy insertion in missions
- Participation in Rover demonstration for IS program and Smart Lander mission

Objectives:

- Detect integration problems early / verification *throughout* lifecycle
- Provide support for module-based verification (scalability)
- Design for verification

Approach:

- Software architecture design and verification
- Module verification with generated assumptions
- Design patterns / architectures for autonomous systems

Accomplishments:

- Modeling and verification of Rover Executive
- Algorithms for module assumptions generation
- Candidate patterns in re-design of Rover Executive

Milestones:

(Jun 02) Demonstrate key capability for de-compositional verification

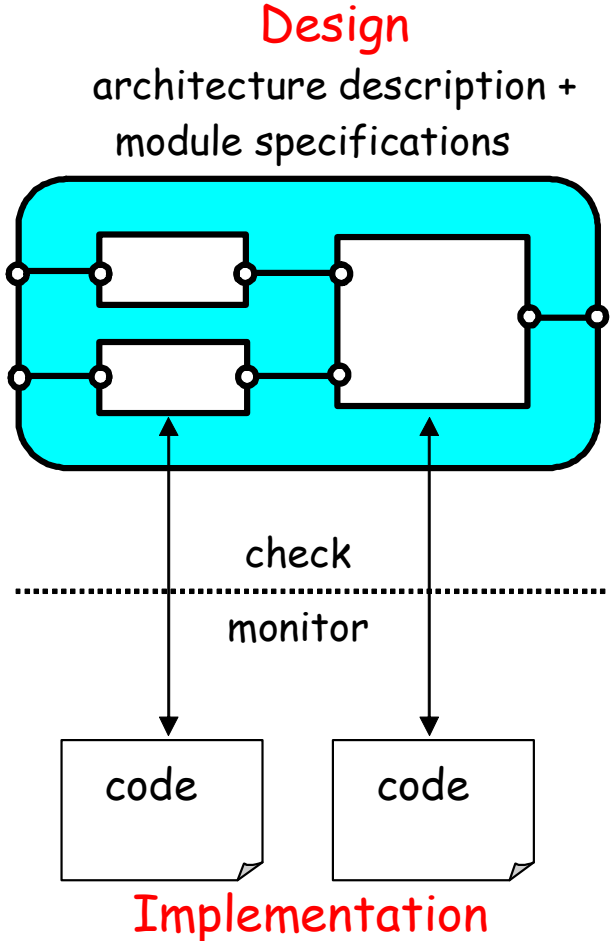
(Sep 03) Demonstrate methodology of combining verification of individual modules into verification at system level

(Mar 04) Demonstrate methodology for scaling up analytic verification to system level on autonomy code

Principal Investigator:
Dimitra Giannakopoulou
NASA Ames Research Center
dimitra@email.arc.nasa.gov

Collaborators:
Corina Păsăreanu, NASA Ames
Richard Washington, NASA Ames

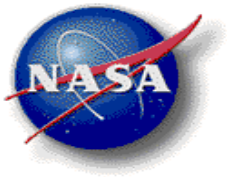
System-Level Verification



- check (system-level) integration properties based on module specifications
- module hierarchy and interfaces used for incremental abstraction
- architectural patterns potentially reusable
- generate module/environment assumptions

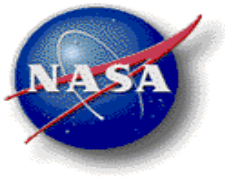
- check implementation modules against their design specifications
- monitor properties that cannot be verified
- monitor environment assumptions

Rover Executive Design



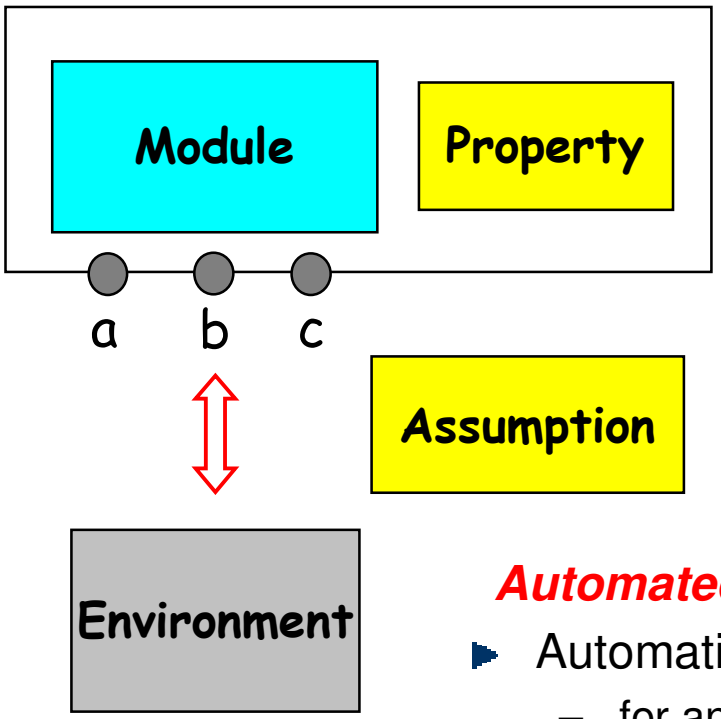
- Systematic translation of the executive design documents into the input language of analysis tool (Labeled Transition System Analyzer - LTSA)
  - Clear correspondence between design notation and model produced
  - Explicit architecture facilitates abstraction and property verification
- Performed verification of the design, with focus on system integration
  - Multi-threaded system; communication through shared variables; synchronization through mutexes and condition variables
  - Properties at this level mainly focus on synchronization: local and global deadlocks, data races, and other related properties specified by the developer
  - Several problems were identified at this level. *Design-level analysis allows for early detection of integration problems*

Verification Results



- Checked race conditions based on design
  - accesses to shared variables are protected by appropriate mutexes
  - detected one data race
- Detected local deadlocks in Executive components
- Investigated a proposed design change by the developer to deal with deadlock
  - model was updated to accommodate the change
  - new local deadlock uncovered a problem with the solution
  - the entire process (model change + analysis) took less than an hour!
- Checked a design decision that the developer was uncertain about
  - automatically decomposed the property across modules; module properties clarified why the decision was correct in the particular design

Module Verification



- Modules may require context information to satisfy a property
- Assumption || Module ⊨ Property
  (assume – guarantee reasoning)

how are assumptions obtained?

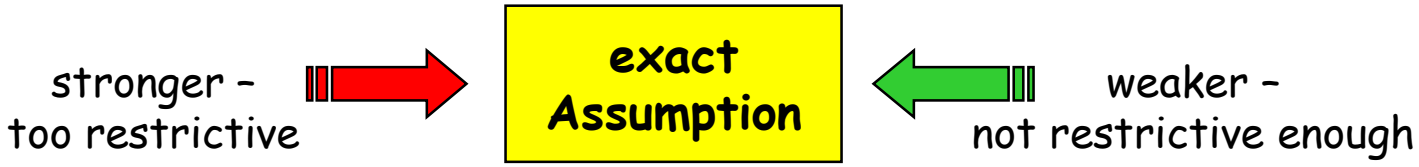
- Developer encodes them
- Abstractions of environment, if known

Automated Software Engineering 2002

- Automatically generate *exact* assumption A
  - for any environment E
    (E || Module ⊨ Property) IFF E ⊨ A
- Demonstrated on Rover example

Assumption Generation

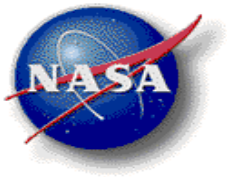
- ASE 2002 algorithm computes Module || Property
  - Module state space may still be too large
  - Generation needs to be incremental
- Algorithms to approach exact assumption incrementally



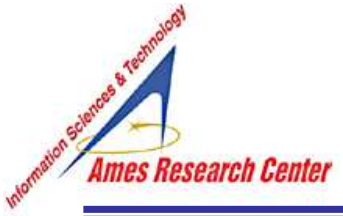
Applications:

- Model checking of modules gives more precise answers
- Support for compositional verification
  - Property decomposition
  - Assumptions for assume-guarantee reasoning
- Assumptions as runtime monitors of environment during deployment

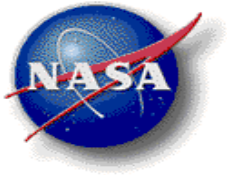
Current Activities



- Develop methodology for combining verification of individual modules into verification at system level
  - Evaluate scalability of algorithms in practice
  - Target specific properties of autonomous systems
- Re-design of Rover Executive
  - early verification of new design
  - investigate leverage from specific design choices for verification
- Plan participation in Rover demonstration for IS program and Smart Lander mission
  - participate in meetings that plan this activity



Design for Verification



Problem – verification as afterthought

- Target systems usually not suitable for direct V&V (size, programming language, missing property specs)
- Requires expensive modelling ("one time effort")
- Gets worse over time (system complexity grows)

Goal – design for verification

- Use specific Design Method to turn V&V into development co-process of target system (ala regression tests)

Solution – methodology and tools

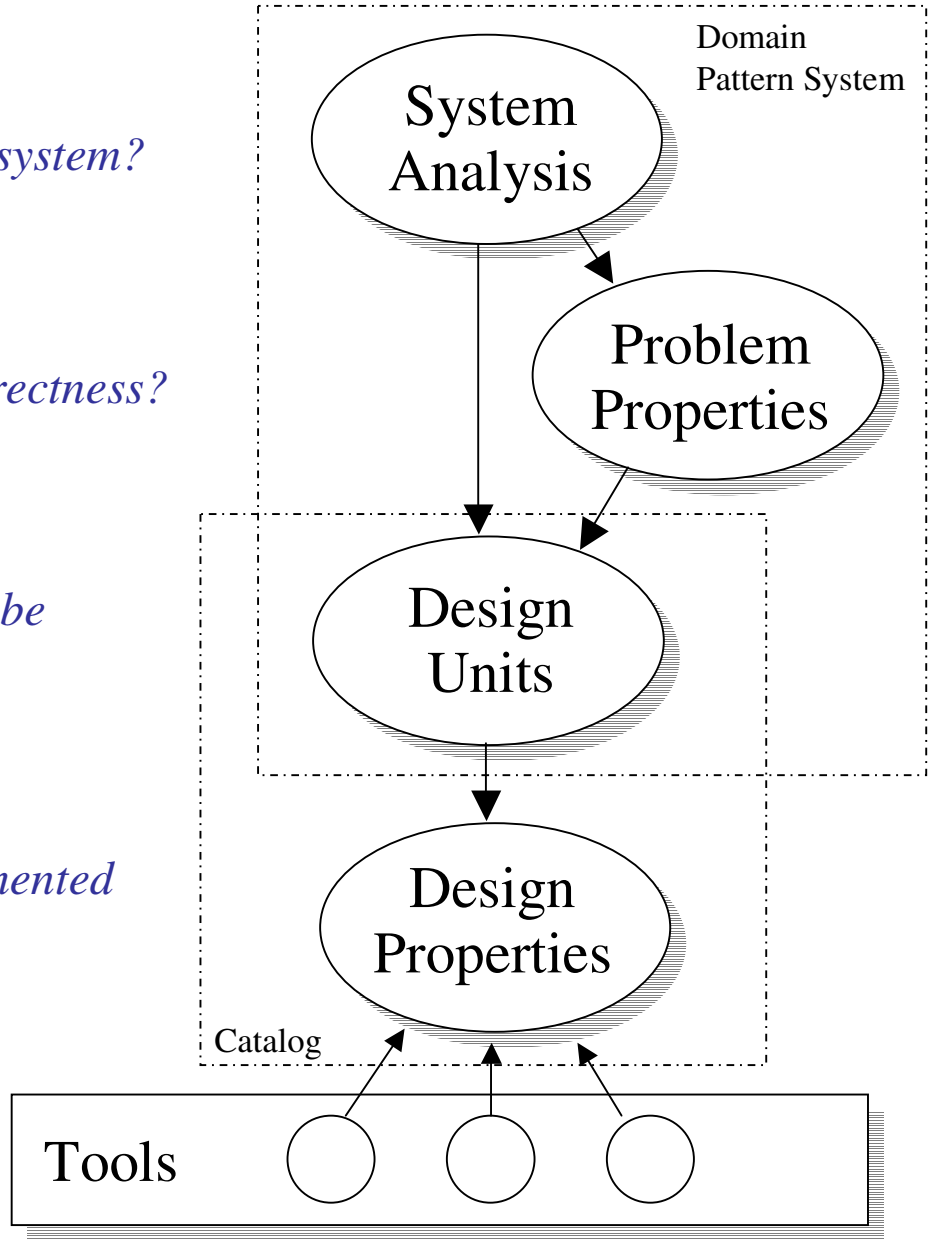
- Derive problem-specific properties from system analysis
- Map into Design Units (concrete Design Patterns capturing single properties) by means of domain specific Pattern Language
- Use Design Unit formal properties to verify implementation

What is the system?

What is correctness?

How can it be achieved?

Is it implemented correctly?



event reaction of
node varies with *node-state*

nodes have to react
gracefully on all *events*
w/o guaranteed event order

Node-internal *Event*
Handlers as *State* objects
(State DP instance)

All *Event* types handled
by every *Node.State* instance

Valid *Node.State* sequences
['init'.. 'end']